Bolt Beranek and Newman Inc.

bbn

LEVEL III

A053 959

12

ɔrt No. 3909

nning for ACCAT Remote Site Operations

Final Report

August 1978

Prepared for:
Defense Advanced Research Projects Agency

12

BBN Report No. 3909

14 BBN-3909

6 Planning for ACCAT Remote Site ~~O~~perations.

9 Final ~~R~~ep[redacted]t. 14 Jun 77-13 Jun 78,

11 Aug[redacted]978

12 30p.

Sponsored by:
Defense Advanced Research Projects Agency
ARPA Order No. 3175.8

Monitored by:
Naval Electronic Systems Command
Contract ~~No.~~ 15 N00039-77-C-0239, ARPA Order-3175
Contract Period: 14 June 1977 to 13 June 1978

Principal Investigator: 10 Robert H./Thomas R./Johnson

060 100 [illegible]

78 09 01 037

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 3909	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Planning for ACCAT Remote Site Operations		5. TYPE OF REPORT & PERIOD COVERED Final Technical 6/14/77 6/13/78
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(S) R. Thomas P. Johnson		8. CONTRACT OR GRANT NUMBER(S) N00039-77-C-0239
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Massachusetts 02138		10. PROGRAM ELEMENT PROJECT TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE August 1978
		13. NUMBER OF PAGES 26
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3175.8.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Advanced Command Control Architectural Testbed (ACCAT)
command control — Remote Site Modules
ARPANET — TENEX Operating System
NSW — interprocess communication
Unix Operating System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes BBN efforts to perform site surveys and planning for the installation of ACCAT remote site modules at selected sites; to provide general system architecture and design services for the ACCAT program; and to assist the ARPA staff in the planning, maintenance, and conducting of demonstrations of various ARPA computer and communication technologies. In addition, it describes the implementation of MSG, an inter-host interprocess communication facility, for the Unix operating system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Planning for ACCAT Remote Site Operations

Final Report

Contents

Summary

The objective of this contract was to perform site surveys and planning for the installation of ACCAT remote site modules at selected sites; to provide general system architecture and design services for the ACCAT program; and, to develop a plan for making selected services of the Fleet Numerical Weather Center (FNWC) available to the ACCAT facility through an FNWC remote site module. In addition, as part of this contract we assisted the ARPA office in the planning, maintenance and conduct of demonstrations of various ARPA information processing technologies.

This project was logically an extension of a previous ACCAT contract which included facility planning for the ACCAT installation at the Naval Ocean Systems Center. Our efforts on ACCAT remote site modules will be continuing through a new contract to procure, install, and maintain remote site modules for a number of sites including the Naval Postgraduate School and CINCPACFLT.

The Advanced Command Control Architectural Testbed (ACCAT) is a facility designed to support evaluation of the applicability of various new computer-communication and information processing techniques to military command and control problems. The ACCAT program is sponsored jointly by ARPA and the Navy.

The core of the ACCAT facility is located at the Naval Ocean Systems Center (NOSC) in San Diego. It began operation in mid-1977. The testbed is built on a number of existing capabilities including: the ARPANET; the ability to provide secure communication for subnetworks within the ARPANET; the standard interfaces and protocols of the network which enable interoperability of heterogeneous equipment; and a large base of existing software and experience in computer networking, time-sharing and interactive computing.

The ACCAT concept includes support for remote site operations. Initially this will involve secure access from distant locations to the core ACCAT facility at NOSC. At a later time, the ACCAT resources may be enhanced with the addition of computing capability at one or more of these remote sites. ACCAT activity at a given remote site will be via a "remote site module" (RSM).

Accomplishments during the period of this contract include the following:

- A site survey for the installation of a remote site module at the Fleet Numerical Weather Center (FNWC) was completed and a site survey report was submitted to ARPA, NAVELEX, and FNWC.

- The requirements for interconnecting FNWC and ACCAT were analyzed to determine the best approach for providing ACCAT access to FNWC meteorological services. A host front-end system developed by Massachusetts Computer Associates for interfacing Air Force computers to the ARPANET was recommended in a report submitted to ARPA, NAVELEX, and FNWC.

- A site survey for installation of a remote site module at the Naval Postgraduate School (NPS) was completed and a site survey report was submitted to ARPA, NAVELEX, and NPS.

- A site survey for installation of a remote site module at CINCPACFLT was initiated. This survey will be completed as part of our contract to install and maintain the CINCPACFLT RSM.

- MSG, the network interprocess communication facility developed to support the National software Works (NSW) system, was implemented for the Unix PDP-11 operating system. With the completion of this implementation all ACCAT hosts support MSG.

- The Resource Sharing Executive (RSEXEC) system, a network operating system originally implemented for ARPANET TENEX hosts, was converted to run under the TOPS-20 operating system.

- We assisted personnel from the ARPA office in a number of demonstrations of ARPA information processing technologies. In addition, we developed several new programs for use in such demonstrations.

The following sections of this report discuss these and other results of our work under this contract in more detail. Further details are to be found in the quarterly technical reports for the contract (BBN Reports 3677 and 38Ø6).

Over the past year a "standard" architecture for ACCAT remote site modules has evolved. Section 2 of this report documents that RSM architecture.

Finally, during the period of this contract we continued to track the progress of the National Software Works system in order to plan for its possible integration into the ACCAT facility. Our recommendations for the role NSW might play in ACCAT are in Section 3 of this report.

1. Site Surveys

During this contract we surveyed sites at the Fleet Numerical Weather Center, the Naval Postgraduate School, and CINCPACFLT that were under consideration for the installation of ACCAT remote site modules.

The objective of a site survey is to plan the preparation of the site for the ACCAT remote site module equipment as well as to plan for installation and operation of the equipment. Generally speaking, the planning activity for a given site adheres to the following pattern. A visit to the site is made to inspect the building that will house the ACCAT equipment and to confer with site personnel. The primary purpose of the visit is to determine how, if at all, the proposed computer area will have to be modified to provide a satisfactory environment for operating the ACCAT equipment. Factors such as the type of floors (e.g., if they are cement, are they sealed properly, are they raised? etc.), walls and ceiling, the availability of power, the amount of air conditioning required, etc. all impact the amount of site preparation required. Shortly after the visit, a site preparation plan detailing such items as environmental and power requirements, equipment layout, cable layout, etc. is prepared. After the plan is reviewed by site and ACCAT personnel there is typically a follow-up meeting, and possibly an additional site visit, to complete the details of the site preparation and installation plan.

The site surveys for the FNWC and NPS remote modules were completed and site survey reports (BBN Reports 3612 and 3746) were submitted to ARPA, NAVELEX, FNWC, and NPS.

A standard ACCAT RSM (See Section 2) will be installed at NPS. Unlike the other remote sites planned for ACCAT, FNWC already has a collection of computers. The purpose of the FNWC remote site module is to make the meteorological services of these computers accessible to ACCAT users. Integration of FNWC into ACCAT will be accomplished by means of a special remote site module. A special host front-end computer (a PDP-11 based system developed by Massachusetts Computer Associates) will be used to connect FNWC CDC-6500 computers to the ACCAT secure subnetwork of the ARPANET (via a Private Line Interface (PLI)).

A survey of the CINCPACFLT site was started as part of this contract. It will be finished under the BBN contract for installation and maintenance of the CINCPACFLT RSM.

2. Standard Remote Site Module Architecture

Over the past year a "standard" architecture for ACCAT remote site modules has evolved. This architecture is the result of cooperative efforts by the Rand Corporation, the Information Sciences Institute of USC, and BBN.

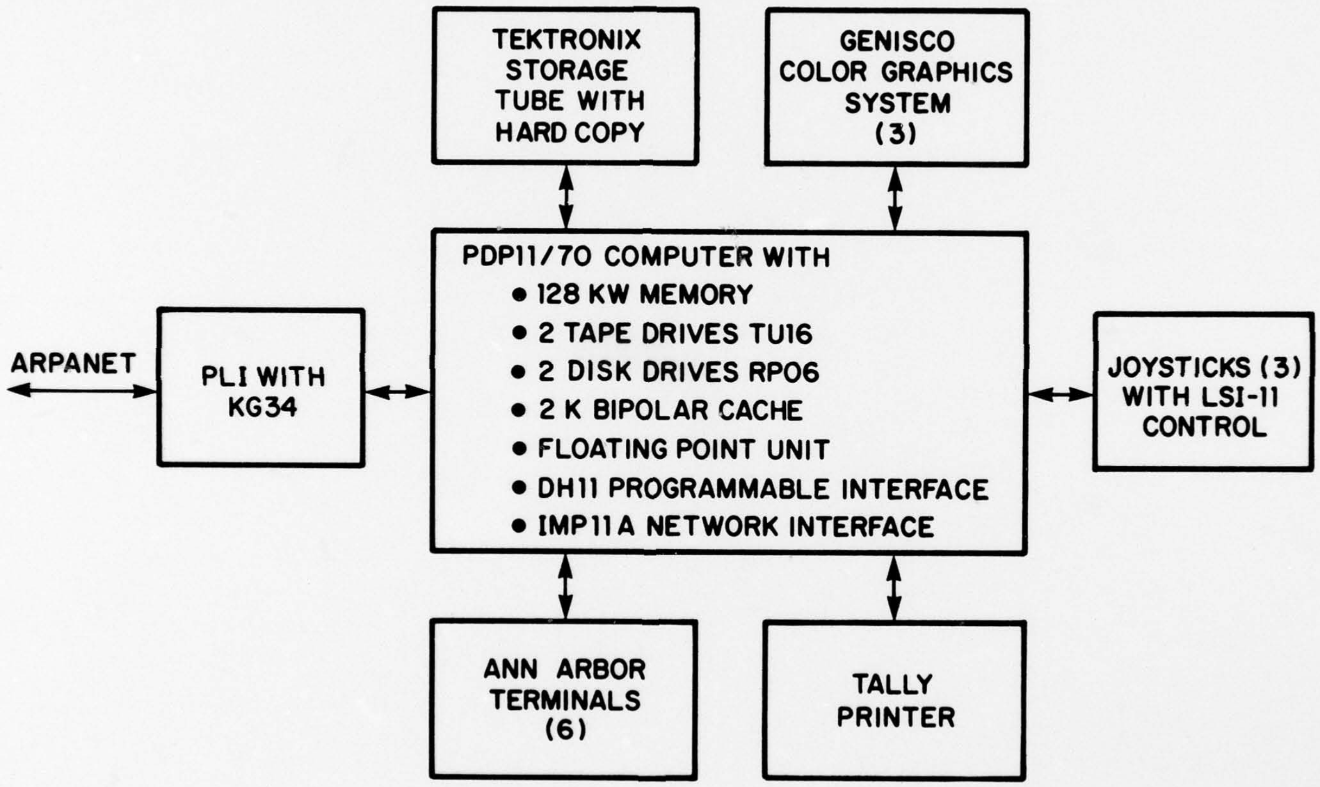
The architecture includes:

- a network host minicomputer (PDP-11/70) with rotating memory, magnetic tape, and Ann Arbor alphanumeric video terminals.
- a three station, interactive, low resolution bit-map color graphics subsystem (based on Genisco hardware) with keyboard and joy-stick input.
- a single high resolution, interactive graphics storage-tube terminal (Tektronix 4014-1).
- a microcomputer (DEC LSI-11) to handle joy-stick input for the minicomputer-graphics system.

Each RSM configuration will be interfaced to the ARPANET via a Private Line Interface (PLI) which will make the RSM a host on the secure ACCAT subnetwork of the ARPANET. The standard RSM architecture is shown schematically in Figure 1.

More specifically, the equipment included in the standard RSM configuration is:

RSM CONFIGURATION STANDARD



Minicomputer:

PDP-11/70-VA computer consisting of:

- 11/70 central processor with memory management
- 2K bipolar cache memory
- 128K byte parity core memory
- Bootstrap/Diagnostic Loader
- Line Frequency Clock
- DECWriter II console terminal
- Terminal control
- Two cabinets (one for CPU, one for core memory)
- Prewired space for mounting additional optional equipment

MJ11-BE 128K bytes core memory

(1 or more) RWP06-AA single access 176 MByte disk drive and control; one disk pack included, expandable to 8 RP drives

TWE16-EA 1600/800 bpi magtape drive with control

FP11-C Floating Point Unit

H960-DH Cabinet with nine SU expander box

DH11-AD Programmable asynchronous 166 line multiplexor

DZ11-AA 8 line asynchronous multiplex for EIA/CCITT terminals or lines

DD11-DK Back panel

IMP11-A Interface for ARPANET IMP

Alphanumeric Video Terminals:

6 Ann Arbor terminals, modified as specified by Rand Corporation

Display Subsystem:

1 GCT-3011 Programmable Graphics Processor

9 GCT-3021-08 Memory Units

2 GCT-3041-60A Chassis and Power Supply

1 GCT-3031 Video Control

3 GCT-3032-3 Monitor Controls

1 GCT-3052 PDP-11 Interface

3 GCT-3084 25" Monitors

3 GCT-3071 Keyboards

3 GCT-3073 Joy-Sticks

9 GCT-3094 BNC

6 GCT-3095 RS232 Cables

1 GCT-3052A Unibus Cable

High Resolution Graphics Terminal:

Tektronix 4014-1 large screen direct view storage display with option-1 communications selector switch.

A particular RSM configuration may include additional equipment; for example, the NPS RSM will include two line printers and a second disk drive.

3. The Role of the National Software Works in ACCAT

The National Software Works (NSW) system is a network operating system being developed under ARPA and Air Force funds. It is designed specifically to provide an effective environment for software production. In particular, NSW provides uniform access to a wide variety of software tools (e.g., application programs) that reside on a heterogeneous collection of host computers on the ARPANET. To use these tools in an integrated fashion NSW users need not be aware of the fact that they reside on different computer systems nor the fact that the data files referenced by the tools may not reside on the same host as the executing tool. Prototype versions of the NSW system have been successfully demonstrated and the first release of the NSW software was made in late 1977.

As part of a previous ACCAT contract (MDA 903-76-C-0211, See BBN Report No. 3582) we investigated the possibility of integrating the NSW system into the ACCAT facility. There are two somewhat different motivations for considering this integration:

- ACCAT is a distributed, heterogeneous, multi-computer facility. At present there is no uniform operating system for the ACCAT facility as a whole; rather, users must deal with each of the constituent host operating systems individually. An operating system which allowed users to deal with the facility as an integrated entity rather than as a collection of autonomous hosts would permit much more effective use of the ACCAT resources.

- The NSW system acts to manage a collection of distributed, heterogeneous resources in a way that provides a uniform, reliable service in the presence of communication network and host computer outages. As such, NSW and its underlying technologies are appropriate technologies for demonstration and evaluation within the testbed.

Our recommendations at the time (May 1977) with regard to the use of NSW within ACCAT can be summarized as follows:

- As of May 1977 the NSW system was neither sufficiently reliable nor adequately responsive to be used in ACCAT as a testbed operating system.

- The MSG facility which was developed to support inter-host communication between NSW system components is useful independent of NSW and should be made accessible in the testbed for use in applications that require inter-host communication.

- The NSW project should continue to be closely tracked since significant performance and reliability improvements are expected over the next year.

Section 4 reports on our efforts to make MSG available in the testbed.

During the period of the current contract we have continued to monitor the status of the NSW project. The following points are some observations and recommendations regarding the possible role of NSW within ACCAT:

- The core ACCAT facility has been in operation over a year. During that period users have of necessity had to deal with the collection of ACCAT hosts without the benefit of a uniform network operating system such as provided by NSW. As a result, users have developed procedures for coping with the distributed, multi-computer nature of the facility. Hence, the urgency of the first motivation for integrating NSW into ACCAT has been considerably diminished.

- MSG will shortly be available on all ACCAT hosts (See Section 4) to support command and control application programs which require inter-host interprocess communication. Since MSG supports some of the desired properties of a uniform testbed-wide system, this serves to further diminish the urgency of the first motivation.

- Despite substantial improvements in NSW performance and reliability it is our opinion that NSW is still not ready to be integrated into ACCAT as a testbed operating system.

- Further improvements in NSW performance and reliability are expected over the next several months. The question of integrating NSW into ACCAT as a testbed operating system should be re-evaluated early in calendar 1979. In our opinion, this re-evaluation must not only consider whether the performance and reliability of NSW are adequate for ACCAT requirements but also the impact its introduction is likely to have on ongoing ACCAT activity. As noted in the first two points above the passage of time has diminished the urgency of the first motivation; users have been using ACCAT for over a year, and the introduction of a new operating system for ACCAT may meet with substantial user resistance, regardless of its technical merits.

- The second motivation for integrating NSW into ACCAT remains valid despite the passage of time. In fact, the testbed currently has no demonstration software which illustrates some of the interesting potentials of distributed processing. The current version of NSW is sufficiently robust and interesting to demonstrate many of these potentials, particularly the notion of a uniform network operating system. Therefore the most recent version of the NSW system should be integrated into the testbed immediately as a demonstration vehicle.

4. Implementation of MSG for UNIX

As noted above, last year we recommended that MSG be installed into ACCAT to support the inter-host interprocess communication requirements of ACCAT application programs. This recommendation was corroborated at a meeting of ACCAT project participants held at the Rand Corporation in July of 1977. At that meeting MSG was established as the standard mechanism for ACCAT interprocess communication.

ACCAT presently includes three different host types: TENEX, TOPS-20, and Unix. As part of our work on the NSW project we have developed MSG implementations for TENEX and TOPS-20 which are suitable for use within ACCAT. However, at the time MSG was established as the ACCAT standard for interprocess communication, no implementation of it for Unix existed. To remedy this situation, as part of this contract we implemented MSG for Unix.

4.1. Status of Implementation

An initial Unix MSG implementation is nearly complete. It will be completed when modifications to the Unix operating system are made to enable the "await" and "capac" operations to work with NCP network connections and with pipes. The implementation provides the same functionality as the TENEX/TOPS-20 MSG (See BBN Report No. 3540) but without the OpenConn and CloseConn operations. These operations require possibly major modification to the Unix NCP and Telnet user and server programs.

4.2. Summary of Implementation Approach

There are three components to the Unix MSG implementation:

. MSG Primitive Interface routines:

These routines are loaded with and called by the user process to invoke MSG primitives. They check all parameters, allocate and set up Pending Event control blocks, communicate with the associated Local MSG (over a pair of pipes), and deliver completed Pending Events.

. Local MSGs:

One of these is a sibling process of each user process that makes use of MSG. A local MSG communicates with the user process via the MSG Primitive Interface routines, passing any requests from the user process on to the appropriate other Local MSGs or the Central MSG, and passing any communications from them on to it as necessary. The Local MSGs do all buffering of pending received messages, and are responsible for timing out Pending Events.

. Central MSG:

This is a collection of processes that support inter-host MSG communication, control certain MSG resources (such as process IDs), and handle the creation of new MSG processes to handle received Generic Messages. Communication with the Central MSG processes is over specially named ports and over network connections.

The operation of a MSG configuration is controlled by two text files: the generic name file and the network configuration file. These specify, respectively, the generic names known to MSG and the network hosts known to MSG. The generic name file defines for each generic process class the correspondence between the class generic name and generic code, the core image file for processes in the generic class, other hosts that may support that generic class, and the start up and termination procedures for processes in that class. The network configuration file specifies the hosts known to MSG and the socket number to use to contact the MSG at that host.

Both of these files, the executable files that comprise MSG, and all ports used by an MSG configuration reside in a single Unix directory: the MSG directory. Access to this directory is restricted to only allow appropriate users to run a program to set up a Local MSG talking to their user program. The Local MSG has access to the MSG directory and thus to other Local MSGs and to the Central MSG, but the user's process does not.

4.3. MSG Primitive Interface Routines

The MSG Primitive Interface routines are implemented in the programming language C, the principal implementation language for Unix. The calling conventions conform closely to the specifications of MSG primitives in Section 2 of the MSG specification document (See BBN Report No. 3483, "MSG: The

Interprocess Communication Facility for the National Software Works").

Since none of the examples of signals in the MSG specification document are appropriate in the Unix environment, a new signal has been provided: the "request signal". For this the user process will, when it has completed whatever else it wishes to do, call the routine RequestSignal(DoWait). If there are no pending events the routine returns immediately with an appropriate response code. If the boolean DoWait is true (-1), the MSG Primitive Interface routines will block until a pending event completes and will then return its event handle. If DoWait is false (0), the routine will return immediately either with the event handle of a completed pending event or, if their are none, with a special response code. The signal parameter code for the request signal is zero.

4.4. The Local MSGs

The Local MSGs act as intermediaries between their associated user process and the rest of an MSG configuration. They perform additional error checking, buffer pending received messages, direct outgoing traffic to the appropriate destination, time out pending events, and protect the rest of an MSG configuration from any misbehavior by the user process.

A Local MSG communicates with the MSG Primitive Interface routines that are part of its associated user process over a pair of pipes. All interaction between them is via messages (similar to the MSG-to-MSG protocol messages) sent over these pipes.

The Local MSG is contacted by other Local MSGs and by the Central MSG through one of two ports. For all traffic directed specifically to the user process, the port used has a name composed of the generic name and process ID of the process. When a ReceiveGeneric is outstanding, the Local MSG will have a port open with a name composed simply of the generic name of the generic class of the user process. The Local (and Central) MSGs are thus able to send traffic directly to the correct destination without needing a special shared database to keep track of processes and generic receives. They are also thus able to use the Unix file system protection mechanisms to prevent interference by other processes.

A Local MSG is started either by the Control process of the Central MSG in response to a received generic message, or by a user who wishes to run a user program in the MSG environment. To do this a special program is run that asks a series of questions to determine the program to run, the generic class it belongs to, and other relevant parameters. A Local MSG process and a user process are then started with a pipe between them, the Local MSG running with the effective user ID of the MSG directory, and the user process running with its normal user ID.

4.5. The Central MSG

The Central MSG is composed of three types of processes:

- . The Control process
- . The Contact process
- . The Network processes

The Control process is generally responsible for controlling the other components of an MSG configuration. It starts up the Contact process, and any Local MSG/user process pairs that the generic name file indicates should be started at initialization. It allocates and hands out process IDs upon request from Local MSGs. It receives Generic Messages for which a receive port does not exist and starts a Local MSG/user process pair to receive the message. It receives any traffic destined to a remote host for which a Network process (see below) does not exist, and starts one up to handle the traffic. All contact with the Control process is through a specially named port.

The Contact process is responsible for the initial interactions with remote MSGs on other hosts that contact this MSG configuration. The MSG contact procedure follows the standard ARPANET Initial Connection Protocol (ICP). The Contact process is normally waiting on the MSG listening contact socket. When a remote MSG completes an ICP exchange involving that socket, the Contact process then goes through the synchronization procedure with it, and, if all is correct, then spawns a Network process to handle further interactions with that host. The Contact process then resumes listening for further contacts.

For each remote host that is in communication with the local MSG configuration there is a Network process responsible for all communication with the MSG system on the remote host. The Network Process uses a pair of network connections to communicate with that MSG and uses a port, named after the host it is in contact with, to communicate with the rest of the local MSG configuration. It is responsible for converting between internal and external formats as necessary, for directing received traffic to the appropriate Local MSG (or to the Control process), and generally for handling the MSG-to-MSG protocol.

5. Software Assistance

As part of the ACCAT program the Computer Corporation of America (CCA) is developing a distributed data base management system called SDD-1 (for System for Distributed Data). SDD-1, which will run on the ACCAT computers, will enable users and user programs to query and update data bases which are distributed among many computers. Since SDD-1 will analyze query and update requests and decompose them into requests upon the proper data base sites, users need not be aware of the distributed nature of the data bases in order to access the data.

CCA has chosen to use the MSG interprocess communication facility to support the inter-host interprocess communication requirements of SDD-1. We have made the TENEX and TOPS-20 implementations of MSG available to CCA. In addition, during the contract period we have consulted with CCA personnel on the use of MSG. This consulting has included teaching CCA how to use MSG effectively, answering questions about MSG, responding to trouble reports, and incorporating suggestions from CCA for improvements into new versions of MSG.

6. Conversion of RSEXEC for TOPS-20

The RSEXEC system is a network operating system which serves as the basis for many ARPA demonstrations. RSEXEC was developed to run under the TENEX operating system. Due to incompatibilities between TENEX and the newer DEC TOPS-20 system, the TENEX version of RSEXEC did not operate properly under TOPS-20 and consequently could not be used on the ACCAT TOPS-20 host. This situation was corrected by modifying the RSEXEC program to operate properly under both operating systems. These modifications were done in a way that allows the same object module for RSEXEC to execute on either host type.

Two problems had to be solved to accomplish the necessary RSEXEC modifications. First, the TOPS-20 implementations for a number of the operating system calls (JSYSs) used by RSEXEC are slightly different from the TENEX implementations for these calls. For example, while functionally equivalent, the TOPS-20 and TENEX implementation of the PMAP and JSYS trap JSYSs have different conventions for passing parameters and returning results. This problem was solved by programming RSEXEC to check the type of host on which it is executing and to then use the JSYS parameter conventions appropriate to that host type.

The second problem derives from the fact that TENEX and TOPS-20 employ slightly different file name syntaxes. In particular, on TENEX semi-colon (;) is used to separate the extension component of a file name from the version number component, while on TOPS-20 period (.) is used. RSEXEC must be prepared to deal with both syntactic conventions, using each when appropriate, since it must be able to run on both host types and since it must be able to cooperate with remote RSEXEC server processes that run on both host types in order to perform file operations. For example, RSEXEC must be able to move the TENEX file named A.B;5 to a TENEX host or to a TOPS-20 host, and to subsequently manipulate it after it is moved; on the TENEX target host the file would be named A.B;5, while on the TOPS-20 target host it would be named A.B.5. There are two aspects to the solution of this problem. First, a user is permitted to use semi-colon and period interchangeably as punctuation between file extensions and version numbers when inputing file names. Second, RSEXEC was modified to check the host type of a remote host prior to interacting with it in order to ensure that the correct syntactic conventions are used in file operations with a server process on that host.

7. ARPA Demonstration Support

The ARPA staff frequently conducts demonstrations designed to illustrate new information processing techniques and concepts that ARPA is investigating, and to present the results of various ARPA programs. Insuring that these demonstrations are effective in illustrating the various technologies and that they proceed as planned with no failures is a time consuming task that requires attention to many details.

As part of this contract we assisted the ARPA office in the presentation of these demonstrations. This demonstration support included maintenance of existing demonstration software, development of new demonstrations and demonstration scenarios, and assistance in the demonstration presentations themselves.

During the contract period we assisted in the presentation of a number of ARPA demonstrations including the following:

- August 1977 demonstration at NOSC in San Diego for the Intelligence Research and Development Board.

- September 1977 demonstration at SRI for Army officials. The focus of this demonstration was the ARPA Packet Radio project.

- December 1977 at ESD at Hanscom Field, Massachusetts for the Air Force on ARPA research programs.

- December 1977 at Augusta, Georgia for the Army Signal Corps. The focus of this demonstration was Packet Radio and ARPA Command Control programs.

- March 1978 in San Diego for the Navy. This demonstration focused on the ACCAT program.

- April 1978 in Stuttgart, Germany and Mons, Belgium. This included a briefing on ARPA developed technologies and planning for a future demonstration at HQEUCOM and SHAPE.

- June 1978 in Augusta, Georgia. This was a BAA III demonstration of Packet Radio and other ARPA developed technologies.

In addition to assisting at the actual demonstrations, we developed several new demonstrations including: a simple interactive program that generates Pascal's triangle; an automatic file transfer program which is used to demonstrate the ability of the ARPANET and its hosts to support file transfers between heterogeneous and geographically separated host computers; and a demonstration of the application of the JANUS system, a relational data base management system that runs on Multics, to a military problem.

Finally, as part of this demonstration assistance effort we periodically exercise the demonstration software to ensure that it remains operational.